Form Approved
OMB No. 0704-0188

RT DATE	3. REPORT TYPE AND DATES COVERED
August 1991	final report 01Sep90-30Sep90

4. TITLE AND SUBTITLE

Open Architectures for Formal Reasoning

5. FUNDING NUMBERS

C: N00039-84-C-0211
T: 26

6. AUTHOR(S)

John McCarthy

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Computer Science Department
Stanford University
Stanford, CA 94305

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

sponsoring agency:
SPAWAR 3241C2
Space & Naval Warfare Systems Command
Washington, D.C. 20363-5100

monitoring agency:
ONR Resident Representative
Mr. Paul Biddle
Stanford Univ., 202 McCullough
Stanford, CA 94305

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release: distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

See attached report.

14. SUBJECT TERMS

15. NUMBER OF PAGES
3

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UL	UL	UL	UL

Sponsored by

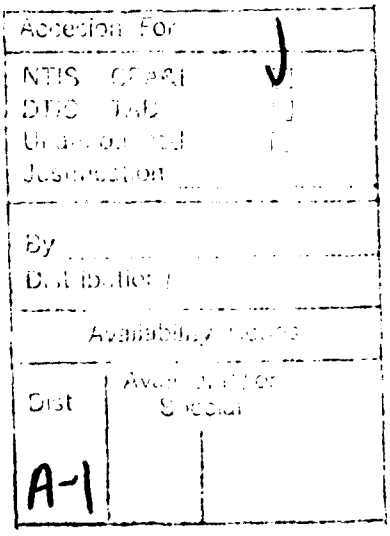
Defense Advanced Research Projects Agency (DoD)
3701 North Fairfax Drive
Arlington, VA 22203-1714

"Open Architectures for Formal Reasoning"

ARPA Order No. ? (can't locate)

Issued by Space and Naval Warfare Systems Command

Under Contract No. N00039-84-C-0211, Task 26



Final Report for Task 26 of Contract N00039-84-C-0211

Open Architectures for Formal Reasoning

Project Summary

This task represents an initial part of a long term project aimed at making both theoretical and practical advances in the field of formal reasoning. The main goal is to provide a framework for designing and experimenting with symbol manipulation programs, and in particular, to provide a general software architecture for implementing formal reasoning systems and interfaces to existing software components including special purpose theorem provers, program transformers, and databases. The kernel will be a computation system that supports a rich collection of data structures for formal reasoning, a wide spectrum of programming paradigms including both high-level and low level constructs, and objects as self contained entities that may be used uniformly and independently of internal representation.

1. Kernel data structures

In [3, 2] we report recent work on a theory of binding structures. Binding structures enrich traditional abstract syntax trees by providing support for representing binding mechanisms and structures with holes. The goal of this work is to establish a common core for building tools such as theorem provers, transformers, static analyzers, evaluators, rewriters, etc. that manipulate symbolic structures. Binding structures solve problems of variable name conflict and renaming, and provide a means for manipulating occurrences of structures. They incorporate the notion of syntactic context. This allows for expression of schemata within the language rather than as meta-expressions. Filling holes is a mechanism for capturing free variables, in contrast to substitution for free variables, which avoids capture. Binding structures provide a basis for sharing a wide range of data structures among program manipulation and mechanized reasoning programs. These include not only terms and formulas, but proofs, rewriting contexts, specifications, etc. In developing efficient programs for manipulation of symbolic structures it is important to be able to express sharing and updating optimizations for algorithms and to have a clear semantics of the structures that support such optimizations. Binding structures together with the work on equivalence of programs that operate on mutable data are a first step in this direction.

Drafts of a full version of the binding structure paper have been distributed to people involved in implementation of theorem provers, programming environments, and program transformation systems (among others) in order to get feed back about possible deficiencies of the theory, and to get suggestions for additional applications. This is also a first step in starting discussions that will hopefully lead to some agreement within the community as to sharable data structures, and a common basis for implementation of symbolic manipulation programs.

2. An Exercise in Verification

In [1] we present a formal verification of the local correctness of a mutex algorithm using the Boyer-Moore theorem prover. The formalization follows closely an informal proof of Manna and Pnueli. The proof method of Manna and Pnueli is to first extract from the program a set of states and induced transition system. One then proves suitable invariants There are two variants of the proof. In the first (atomic) variant, compound tests involving quantification over a finite set are viewed as atomic operations. In the second (molecular) variant, this assumption is removed, making the details of the transitions and proof somewhat more complicated.

The original Manna-Pnueli proof was formulated in terms of finite sets. This led to a concise and elegant informal proof, however one that is not easy to mechanize in the Boyer-Moore logic. In the mechanized version we use a dual isomorphic representation of program states based on finite sequences. Our approach was to outline the formal proof of each invariant, making explicit the case analyses, assumptions and properties of operations used. The outline served as our guide in developing the formal proof. The resulting sequence of events follows the informal plan quite closely. The main difficulties encountered were in discovering the precise form of the lemmas and hints necessary to guide the theorem prover.

The complete formal proofs (input to the Boyer-Moore prover) appear as appendices. Some comments on formalization techniques, difficulties, and alternatives are included as comments in the theorem prover input.

3. Surveys

A first step in designing a architecture for formal reasoning and other symbolic manipulation systems is to survey and analyze existing systems and technologies. Work is in progress on two surveys: one of existing theorem provers, and one of programming environment kernels and tools. In both cases a major goal is analyzing the common components that implementations could share. In the case of programming environments we also want to determine to what degree existing systems support sharing and interoperability, and what is needed to make the mechanisms more widely accepted. As part of the theorem prover survey we also want to determine capabilities of existing systems – the language, logic, proof-theory, proof-mechanisms, means of interaction with a user, major applications, and existence of tutorials. The resulting data is intended to help potential users find the system that best meets their needs. It will also be used to develop requirements for general purpose, mechanized reasoning systems capable of supporting a wide range of non-trivial applications.

4. References

[1] Misao Nagayama and Carolyn Talcott. An nqthm mechanization of "an exercise in the verification of multi-process programs". Technical Report to appear, Computer Science Department, Stanford University, 1991.

[2] Carolyn L. Talcott. Binding structures. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation*. Academic Press, 1991.

[3] Carolyn L. Talcott. Towards a theory of binding structures. In *Second International Conference on Algebraic Methodology and Software Technology, AMAST*, 1991.